Methods to:

Query for information

In this lecture, we will go through some methods from Standard Ruby Library.
You will learn how to query classes about class itself, its' methods and variables.

Querying info about Classes

```ruby
module MyModule
end

class Parent
    def foo
    end
end

class Child < Parent
    include MyModule
end

obj = Child.new

foo = obj.class                                  # Child
foo.name                                         # "Child"

Child.ancestors                                  # [Child, MyModule, Parent, Object, Kernel, BasicObject]
Child.included_modules                           # [MyModule, Kernel]
Child.singleton_class?                           # false
Child.singleton_class.singleton_class?           # true
Child.include? MyModule                          # true

obj.instance_of? Child                           # true
obj.instance_of? Parent                          # false

obj.kind_of? Child                               # true
obj.kind_of? Parent                              # true
obj.kind_of? MyModule                            # true
```

Querying info about Methods

```ruby
module MyModule
    def module_method
    end
end

class Parent
    def parent_method
    end
end

class Child < Parent
    include MyModule

    def child_method
    end
end

obj = Child.new

def obj.singleton_method
end

obj.methods                                 # [:singleton_method, :child_method, :module_method, :parent_method, :nil?, :===, :=~, :!~, :eql?
obj.methods(false)                          # [:singleton_method]
obj.singleton_methods                       # [:singleton_method]

Child.instance_methods                      # [:child_method, :module_method, :parent_method, :nil?, :===, :=~, :!~, :eql?, :hash, :<=>, :cla
Child.instance_methods(false)               # [:child_method]

Child.method_defined? :singleton_methods    # true
Child.method_defined? :child_methods        # true
Child.method_defined? :parent_methods       # true
Child.method_defined? :module_methods       # true
Child.method_defined? :clone                # true

obj.respond_to? :child_methods              # true
obj.respond_to? :clone                      # true


obj.singleton_class.instance_methods(false) # [:singleton_method]
```

Querying info about Variables

```ruby
$global_variable = "global"

TOP_LEVEL_CONST = "top level constant"

class Parent
    @@parent_class_var = "parent class variable"

    PARENT_CONST = "constant in Parent class"

    @parent_eigenclass_instance_var = "eigenclass instance variable"
end

class Child < Parent
    @@child_class_var = "child class variable"

    CHILD_CONST = "constant in Child class"

    @child_eigenclass_instance_var = "eigenclass instance variable"

    def initialize
        @child_class_instance_var = "Child instance variable"
    end

    class InnerClass
    end
end

obj = Child.new
```

```ruby
obj.instance_variables                                  # [:@child_class_instance_var]
obj.instance_variable_defined? :@child_class_instance_var   # true

Child.instance_variables                                # [:@child_eigenclass_instance_var]
Parent.instance_variables                               # [:@parent_eigenclass_instance_var

Child.class_variables                                   # [:@@child_class_var, :@@parent_cla
Parent.class_variables                                  # [:@@parent_class_var]
Child.class_variable_defined? :@@child_class_var        # true

global_variables                                        # [:$global_variable, :$;, :$-F, :$@
local_variables                                         # [:obj]

Child.constants                                         # [:CHILD_CONST, :InnerClass, :PAREN
self.class.constants                                    # [:TOP_LEVEL_CONST, :Parent, :Child

obj.instance_variable_get :@child_class_instance_var    # "Child instance variable"

obj.instance_variable_set(:@child_class_instance_var, 'changed value')
obj.instance_variable_get :@child_class_instance_var    # "changed value"

Child.class_variable_get :@@child_class_var             # "child class variable"

Child.const_get :PARENT_CONST                           # "constant in Parent class"
Child.const_get :InnerClass                             # Child::InnerClass
```